CS112 - Fall 2022

Lab13

Instructor: Paul Haskell

INTRODUCTION

This week you will install the Eclipse Java debugger. Then you can use it to write and debug several programs, which you will turn in.

Installing Eclipse

Point a browser to https://www.eclipse.org/downloads/ and download the "Eclipse IDE 2022-06". Save the downloaded file somewhere you can find it. Run the installer, and if asked, install the "Eclipse IDE for Java Developers". Pick a folder where Eclipse should be installed. I have **not** had good look installing it in the usual "Program Files" directories. I put it into a personal directory or Temp directory.

Eclipse will ask you to set up a Workspace where it saves its projects. Please pick a separate directory from what you have been using for class. You should copy files back and forth between your original CS112 directory and the Eclipse Workspace directory, but because of differences where javac and Eclipse store compiled .class files, they can confuse each other if they operate in the same directory.

The Eclipse Welcome Window will ask if you want an overview, tutorial, set preferences, etc. Feel free to do those on your own as you want to learn more about Eclipse. For now, you can just click the "Hide" button.

Eclipse Basics

When you start Eclipse, you can create a new Java Project by using the File->New menu to select the "Java Project" option. Do not create a module—select the "Don't Create" option when asked.

Find your new project in the list of projects on the left side of the Eclipse window. Open the little pulldown menu to see the "src" directory. Right-click that and enter a Name for your new Java source file (you don't have to put it into a package right now). Now the source code window should open; this lets you edit and save the source code right in the window.

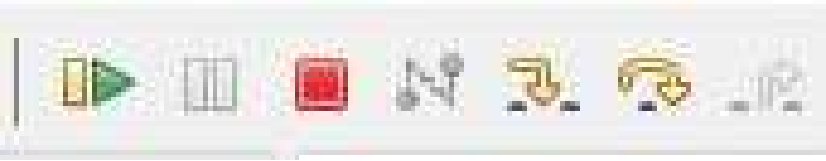
Go ahead and write a basic "Hello World" program. You can compile and run it in one step by clicking the green circle with the white triangle inside.

You might have noticed that Eclipse automatically highlights Java errors—and sometimes even suggests fixes—in the source code window. Very convenient.

Debugging with Eclipse

Add a few more lines to your Hello World program, maybe a few more printouts or a simple loop. On the left side of the source code window there are line numbers. Right-click a line number that has an actual statement on it and select "Toggle Breakpoint". Then from the Run menu, select Debug. The

program should run up to your breakpoint and then pause. Just below the menu-bar near the top of the screen, some controls appear that look like:



These controls let you:

- Continue from a breakpoint
- Stop the current program execution
- "step into" – if the current statement is a function call, step into the function and pause
- "step through" – if the current statement is a function call, execute the function entirely and pause at the next statement in the source code
- "step return" – complete running the rest of the current function and return to the calling function

You can create more than one breakpoint. Give it a try!

Probably already you can imagine that a graphical debugger like Eclipse will make it a lot easier to figure out what is happening as a program executes.

Feel free to use Eclipse for the rest of the semester. However, when you turn in programs for labs or assignments, they are still expected to compile and run correctly using the javac and java tools.

String Reversal

In **CourseInfo/Week13** is a file called **ReverseString.java** . It has a method that is supposed to take a `String` and convert it to an array of chars, with the chars in the reverse order of the `String`. Please embed this function in a `class ReverseString`, add a `main` function to read a `String` from `args[0]`, reverse the `String`, and print the result. And debug the function of course!

Character Counting

And now a program for you to write on your own, one that we will use as part of the second big class project. Write a program called **CountChars.java** that reads characters from the keyboard, counts how many of each character value occur in the input, and when the keyboard input finishes, prints out a table of the character values that have >0 occurrences and the number of occurrences. So your sample output might look like:

A 13
B 4
C 8
…
a 48
b 11
etc

Let's only consider characters with Unicode values from 0 through 255. If the input has any characters with values greater than 255, the program should ignore them. (And I might test with them!)

We won't worry about counting the actual Newline characters in the input ('\u000a' and '\u000d').

For testing, how do you end keyboard input? You can test with redirection from a file of course. But if you are typing from an actual keyboard,

- On Linux and Mac, you can type CTRL-d (hold down Control key and then press 'd')
- On Windows, you can type CTRL-z and then ENTER

These keyboard sequences generate the End-Of-File (EOF) character, which ends the input.

Reminder

Put all your files in Lab13 and push to GitHub before the deadline. This assignment must be turned in before Friday October 14th

Conclusion

These programs hopefully give you some experience with Eclipse, which you might find useful when we start our first big project next session.

Grading Rubric

ReverseString.java is worth 10 points: 2 points for each of 5 test cases.[1]

CountChars.java is worth 10 points: 2 points for each of 5 test cases.

[1] Earlier version of this doc had an error: not **ReverseChars** but **ReverseString**